AUSTRALIA

jc398 U.S. PTO
09/112777
07/10/98

#2

Patent Office
Canberra

I, MARIO PERUSSICH, ASSISTANT DIRECTOR PATENT SERVICES, hereby certify that the annexed is a true copy of the Provisional specification in connection with Application No. PO 8018 for a patent by SILVERBROOK RESEARCH PTY LTD filed on 15 July 1997.

I further certify that the annexed specification is not, as yet, open to public inspection.

CERTIFIED COPY OF PRIORITY DOCUMENT

WITNESS my hand this Twenty-second day of June 1998

MARIO PERUSSICH
ASSISTANT DIRECTOR PATENT SERVICES

COMMONWEALTH OF AUSTRALIA
AUSTRALIA
PATENT OFFICE

P/00/009
Regulation 3.2

AUSTRALIA

Patents Act 1990

PROVISIONAL SPECIFICATION

Application Title: Image Processing Method and Apparatus (ART24)

The invention is described in the following statement:

GH REF: 23975K

IMAGE PROCESSING METHOD AND APPARATUS (ART24)

Field of the Invention

The present invention relates to an image processing method and apparatus and, in particular, discloses a Producing Automatic "Van Gogh" Effects in Images.

The present invention further relates to the field of image processing and in particular to producing artistic effects in images.

Background of the Invention

Recently, it has become quite popular to provide filters which produce effects on images similar to popular artistic painting styles. These filters are designed to take an image and produce a resultant secondary image which appears to be an artistic rendition of the primary image in one of the artistic styles.

One extremely popular artist in modern times was Vincent van Gogh. It is a characteristic of art works produced by this artist that the direction of brush strokes in flat areas of his paintings strongly follow the direction of edges of dominant features in the painting. For example, his works entitled "Road with Cypress and Star", "Starry Night" and "Portrait of Doctor Gachet" are illustrative examples of this process.

It would be desirable to provide a computer algorithm which can automatically produce a "van Gogh" effect on an arbitrary input image.

Summary of the Invention

It is an object of the present invention to produce automatic "van Gogh" type effects in images.

In accordance with the first aspect of the present invention there is provided a method of automatically processing an image comprising locating within the image features having a high spatial variance and stroking the image with a series of brush strokes emanating from those areas having high spatial variance.

Preferably, the brush strokes have decreasing sizes

near important features of the image.

Additionally, the position of a predetermined portion of brush strokes can undergo random jittering.

Brief Description of the Drawings

Notwithstanding any other forms which may fall within the scope of the present invention, preferred forms of the invention will now be described, by way of example only, with reference to the accompanying drawings which:

Fig. 1 illustrates the major steps in the preferred embodiment;

Fig. 2 illustrates the Sobel filter co-efficients utilised within the preferred embodiment;

Figs. 3 & 4 illustrate the process of offsetting curves utilised in the preferred embodiments;

Description of the Preferred and Other Embodiments

The preferred embodiment is preferable implemented through suitable programming of a hand held camera device such as that described in Australian Provisional Patent Application entitled "Image Processing Method and Apparatus (ART01)" filed concurrently herewith by the present applicant the content of which is hereby specifically incorporated by cross reference.

The aforementioned patent specification discloses a camera system, hereinafter known as an "Artcam" type camera, wherein sensed images can be directly printed out by an Artcam portable camera unit. Further, the aforementioned specification discloses means and methods for performing various manipulations on images captured by the camera sensing device leading to the production of various effects in any output image. The manipulations are disclosed to be highly flexible in nature and can be implemented through the insertion into the Artcam of cards having encoded thereon various instructions for the manipulation of images, the cards hereinafter being known as Artcards. The Artcam further has significant onboard processing power by an Artcam Central Processor unit (ACP)

which is interconnected to a memory device for the storage of important data and images.

In the preferred embodiment there is described an algorithm which will automatically convert a photographic image into a "painted" rendition of that image which replaces groups of pixels in the input image with "brush strokes" in the output image. The algorithm works by automatically detecting dominant edges and propagating the edge direction information into flat areas of the image so that brush strokes can be oriented in such a way as to approximate the van Gogh style. The algorithm is suitable for implementation on the aforementioned Artcam device.

Turning initially to Fig. 1, the algorithm comprises a number of steps 1. These steps include an initial step of filtering the image to detect its edges 2. Next, the edges are thresholded or "skeletonised" 4 before being processed 5 to determine the final edges 6. Bézier curves are then fitted to the edges. Next, the curves are offset 7 and brush strokes are placed on final image 8. The process 7 and 8 is iterated until such time as the image is substantially covered by brush strokes. Subsequently, final "touching up" 9 of the image is performed.

Turning now to describe each step in more detail. In the first step 2 of filtering to detect edges, a Sobel 3 x 3 filter having co-efficient sets 12 and 13 as illustrated in Fig. 2 can be applied to the image. The Sobel filter is a well known filter utilised in digital image processing and its properties are fully discussed in the standard text "Digital Image Processing" by Gonzalez and Woods published 1992 by the Addison - Wesley publishing company of Reading, Massachusetts at pages 197-201. The Sobel derivative filter can be applied by either converting the image to greyscale before filtering or filtering each of the colour channels of an image separately and taking the maximum. The result of Sobel filtering is the production of a greyscale image indicating the per-pixel edge strength of the image.

Next, the resultant per-pixel edge strength image is thresholded 3 so as to produce a corresponding thresholded binary image. The threshold value can be varied however, a value of 50% of the maximum intensity value is suitable. For each pixel in the edge strength image the pixel is compared with the threshold and if it is greater than the threshold a "one" is output and if it is less than the threshold a "zero" is output. The result of this process is to produce a threshold edge map.

Next, the thresholded edge map is "skeletonised" at step 4 of Fig. 1. The process for skeletonising an image is fully set out in the aforementioned reference text at pages 491-494 and in other standard texts. The process of skeletonisation produces a "thinned" skeletonised edge map maintaining a substantial number of characteristics of the thresholded edge map.

In a next step the edges of the skeletonised edge map are determined to yield a data structure which comprises a list of further lists of points within the image. Preferably, only edges having a length greater than a predetermined minimum are retained in the list.

As the skeletonised image contains only single-pixel-width edges, possibly with multiple branches, the following algorithm expressed as a C++ code fragment sets out one method of determining or identifying the points which belong to each contiguous edge in the skeletonised image. It breaks branching edges into separate edges, and chooses to continue along the edge in the direction which minimises the curvature of each branch - ie. at a branch-point it favours following the branch which induces the least curvature. The code is as follows:

```
void
FollowEdges
(
    Image& image,
    int minimumEdgeLength,
```

```
        PointListList& pointListList
)
(
        pointListList.Erase();
        for (int row = 0; row < image.Height(); row++)
        {
                for (int col = 0; col < image.Width(); col++)
                {
                        If (image[row][col] > 0)
                        {
                                PointList pointList;

                                // append the starting point to the point list,
                                // and clear it so we don't find it again
                                pointList.Append(Point(col, row));
                                image[row][col] = 0;

                                // follow the edge from the starting point to its beginning
                                FollowEdge(row, col, image, pointList);

                                // reverse the order of the points accumulated so far,
                                // and follow the edge from the starting point to its end
                                pointList.Reverse();
                                FollowEdge(row, col, image, pointList);

                                // keep the point list only if it's long enough
                                if (pointList.Size() >= minimumEdgeLength)
                                pointListList.Append(pointList);
                        }
```

```
          }
     }
}

// table of row and column offsets to eight surrounding
neighbours
// (indexed anti-clockwise, starting east)
static int offsetTable[8][2] =
{
     {0, 1}, {-1, 1}, {-1, 0}, {-1, -1}, {0, -1}, {1, -1},
{1, 0}, {1,1}
};

// table of preferred neighbour checking orders for given
direction
// (indexed anti-clockwise, starting east favouring non
diagnals)
static int nextDirTable[8][8] =
{

     {0,  2,   6,   1,   7,   3,   4,   5),
     {2,  0,   1,   3,   7,   4,   5,   6),
     {2,  4,   0,   3,   1,   5,   6,   7},
     {4,  2,   3,   5,   1,   6,   7,   0),
     {4,  6,   2,   5,   3,   7,   0,   1},
     {6,  4,   5,   7,   3,   0,   1,   2),
     {6,  0,   4,   7,   5,   1,   2,   3},
     {0,  6,   7,   1,   5,   2,   3,   4),
};

void
FollowEdge
(
     int row,
     int col,
     Image& image,
```

```
        PointList& pointList
    )
    {
        Vector edgeHistory[EDGE_HISTORY_SIZE];
        int historyIndex = 0;

        for (;;)
        {
            // table of pre-computed
            // compute tangent estimate from edge history
            Vector tangent;
            for (int i = 0; i < EDGE_HISTORY_SIZE; i++)
                tangent += edgeHistory[i];

            // determine tangent angle and quantize to eight
directions
            // (direction zero corresponds to the range -PI/8
to +PI/8, i.e east)
            double realAngle = tangent.Angle();
            int angle = (int) ((realAngle * 255) / (2 * PI) +
0.5);
                int dir = ((angle - 16 + 256) % 256) / 32;

                // try surrounding pixels, fanning out from
preferred
                // (i.e. edge) direction
                int* pNextDir = nextDirTable[dir];
                bool bFound = false;

            for (i = 0; i < 8; i++)
            {
                    // determine row and column offset for
current direction
                    int rowOffset = offsetTable[dir][0];
                    int colOffset = offsetTable[dir][1];
```

```
                    //  done  testing  neighbours  if  edge
pixel found

                    if  (image  [row  +  rowOffset]  [col  +
colOffset] > 0)
                    {
                        // determine edge pixel address
                        Point oldPoint (col, row);
                        row += rowOffset;
                        col += colOffset;
                        Point newPoint (col, row);

                        // update edge tangent history
                        tangent = newPoint - oldPoint;
                        tangent.Normalize();
                        edgeHistory[histroyIndex]          =
tangent;
                        historyIndex  =  (historyIndex  +  1)
% EDGE_HISTORY_SIZE;

                        // append edge pixel to point list
                        pointList.Append(newPoint);

                        // clear  edge  pixel,  so  we  don't
find it again
                        image[row][col] = 0;
                        bFound = true;
                        break;
                    }
                    // determine next direction to try
                    dir = pNextDir[i];
                }

                //  done  following  edge  if  no  edge  pixel
found
                if (!bFound)
```

```
                break;
            }
    }
```

The result of utilising this algorithmic component on the skeletonised edgemap is to produce a list of edges having at least a predetermined size. A suitable size was found to be a length of 20 pixel elements.

In the next step 6 of Fig. 1, Bézier curves are fitted to each of the edge lists derived from step 5. For each list of edges, a piece wise Bézier curve is fitted to the corresponding list of points. A suitable algorithm for fitting the piece wise Bézier curve is Schneider's curve fitting algorithm as set out in Schneider, P.J., "An Algorithm for Automatically Fitting Digitised Curves", in Glassner,A.S. (Ed.), Graphics Gems, Academic Press, 1990. This algorithm provides quick convergence to a good fit which aims only for geometric continuity and not parametric continuity. Schneider's algorithm is recursive, such that if the fit is poor, is sub-divides the curve at the point of maximum error and fits the curves to the two halves separately. Next an estimate of the tangent at the split point is derived using only the two points on either side of the split point. For dense point sets, this tends to amplify the local noise. An improved quality of curve fitting can be alternatively undertaken by using points further away from the split point as the basis for the tangent.

In the next steps 7 of Fig. 1, the curves are offset from the primary curve list by half a desired "brush stroke width". The offsetting occurring on both sides of the primary curve list with the result being two curves approximately one stroke width apart from one another which run parallel to and on either side of the original primary curve.

The following algorithm is utilized to generate a piece

wise Bézier curves which are approximately parallel to a specified piece wise Bézier curves and includes the steps.

i. Create an empty point list.

ii. Create and empty tangent (vector) list.

iii. Evaluate selected points on each curve segment making up the piece-wise curve and offset them by the specified offset value. Append the offset points to the point list, and their corresponding tangents to the tangent list. This process is described below with reference to Fig. 2 and 3.

iv. Fit a piece-wise Bézier curve to the resultant point list. Rather than estimating tangents during the curve-fitting process, use the exact tangents associated with the offset points.

Offset each curve segment as follows:

i. Evaluate the curve value, normalised tangent and normalised normal normalised to the size of the image for a set of evenly-spaced parameter value between (and including) 0.0 and 1.0 (eg. a spacing of 0.25).

ii. Scale the normals by the specified offset value.

iii. Construct line segments using the curve points and scaled normals.

iv. If any two line segments intersect, eliminate the point associated with one of them.

v. Append the surviving points to the point list, and append their corresponding tangents to the tangent list. Only append the point associated with parameter value 1.0 if the segment in question is the last in the piece-wise curve, otherwise it will duplicate the point associated with parameter value 0.0 of the next segment.

The process of offsetting each curve segment can proceed as following:

1. Firstly, for a set of evenly spaced parameter values on the Bézier curve between (and including) 0.0 and 1.0, for each parameter value PN (Fig. 3) the curve value 30 a normalised tangent 31 and normalised normal 32 are

calculated.

2. Next, the normals 32 are scaled 34 by a specified offset value.

3. Next a line segment from the point 30 to a point 36, which is at the end of the scaled normal 34 is calculated.

4. Next, the line segment 30, 36 is checked against corresponding line segments for all other points on the curve eg. 38, 39. If any two line segments intersect, one of the points 36 is discarded.

5. The surviving points are appended to the point list and their corresponding tangents are appended to the tangent list. The point associated with the parameter value 1.0 is appended only if the segment in question is the last in the piece-wise curve segment. Otherwise, it will duplicate the point associated with the parameter value 0.0 of the next segment.

Turning to Fig. 4, the end result of the offset of curves in accordance with step 7 of Fig. 1 is to produce for a series of Bézier curve segments C1, C2 etc. Firstly, a series of parametrically spaced points, P1 - P5. Next, the normalisation points N1 - N5 are produced (corresponding through to point 36 of Fig. 3), for each of the points P1 - P5. Next, the resultant piece-wise Bézier curve segment 40 is produced by utilising the points in 1 - N5. This process is then repeated for the opposite curve comprising the points N6 - N10 and curve 41. This process is then repeated for each of the subsequent piece-wise curves C2 etc. The result is the two curves of 40, 41 being substantially parallel to one another and having a spaced apart width of approximately one brush stroke.

Next, a series of brush strokes are placed into the output image along the curves. The strokes are oriented in accordance with the curve tangent direction. Each brush stroke is defined to have a foot print which defines where it may not overlap with other brush strokes. A brush stroke

may only be place along the curve if its foot print does not conflict with the foot prints already present in the output image. Any curves that do not have any brush strokes placed along them are discarded and the process of steps 7 and 8 are iterated in a slightly modified form until no curves are left. The slightly modified form of step 7 is to offset the curves by one brush stroke in the outward direction rather than the half brush stroke necessary when offsetting curves from the curve C1 of Fig. 4.

It has been found by utilisation of the above method that the result produced consists of a series of brush strokes which emanate from objects of interest within the image.

Subsequent to covering the image with brush strokes of a given size, further processing steps can be undertaken with smaller and smaller brush strokes and increasing derivative threshold levels so as to more accurately "brush stroke" important features in the image. Such a technique is similar to that used by van Gogh in certain portions of his images where details are required.

It would be appreciated by a person skilled in the art that numerous variations and/or modifications may be made to the present invention as shown in the specific embodiment without departing from the spirit or scope of the invention as broadly described. The present embodiment is, therefore, to be considered in all respects to be illustrative and not restrictive.

The present provisional is one of a series of Australian Provisional Patent Applications which relate to a new form of technology for the production of images. These Australian Provisional Patent Applications encompass a broad range of fields and as such, the present provisional is best viewed in the overall context of the development of this new form of technology. Appendix A attached hereto sets out the details of each of the series of Australian Provisional Patent Applications and, to the extent necessary, the

associated Australian Provisional Patent Applications are hereby incorporated by cross-reference.

We Claim:

1. A method of automatically processing an image comprising locating within the image features having a high spatial variance and stroking the image with a series of brush strokes emanating from those areas having high spatial variance.

2. A method as claimed in claim 1 wherein said brush stroke have decreasing sizes near important features of the image.

3. A method as claimed in any previous claim wherein said brush strokes include opacity and bump maps for added realism.

4. A method as claimed in any previous claim wherein the position of a predetermined portion of brush strokes undergoes random jittering.

Dated this 15th day of July 1997

Silverbrook Research Pty Ltd
By their Patent Attorneys
GRIFFITH HACK

Abstract

Method is disclosed for the automatic creation of images in a "van Gogh" style. The method comprises locating portions of detail in an image an utilising the areas of detail to propagate brush strokes into areas of the image having lesser details. A number of modifications are also proposed including utilising refining brush strokes to process those areas of detail in an image.

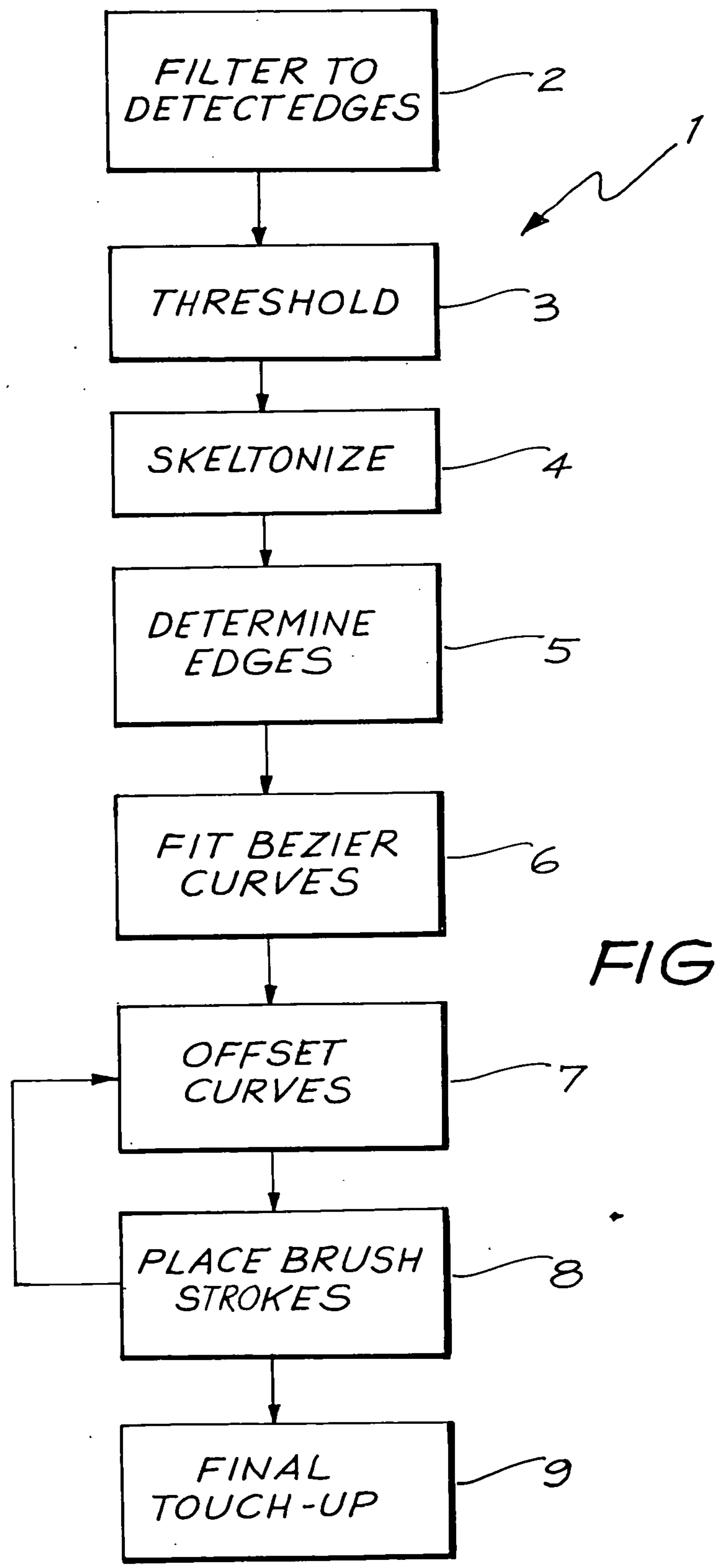
FILTER TO DETECT EDGES
2
1
THRESHOLD
3
SKELTONIZE
4
DETERMINE EDGES
5
FIT BEZIER CURVES
6
OFFSET CURVES
7
PLACE BRUSH STROKES
8
FINAL TOUCH-UP
9

FIG. 1

12

-1	2	-1
0	0	0
1	2	1

13

-1	0	1
-2	0	2
-1	0	1

FIG.2

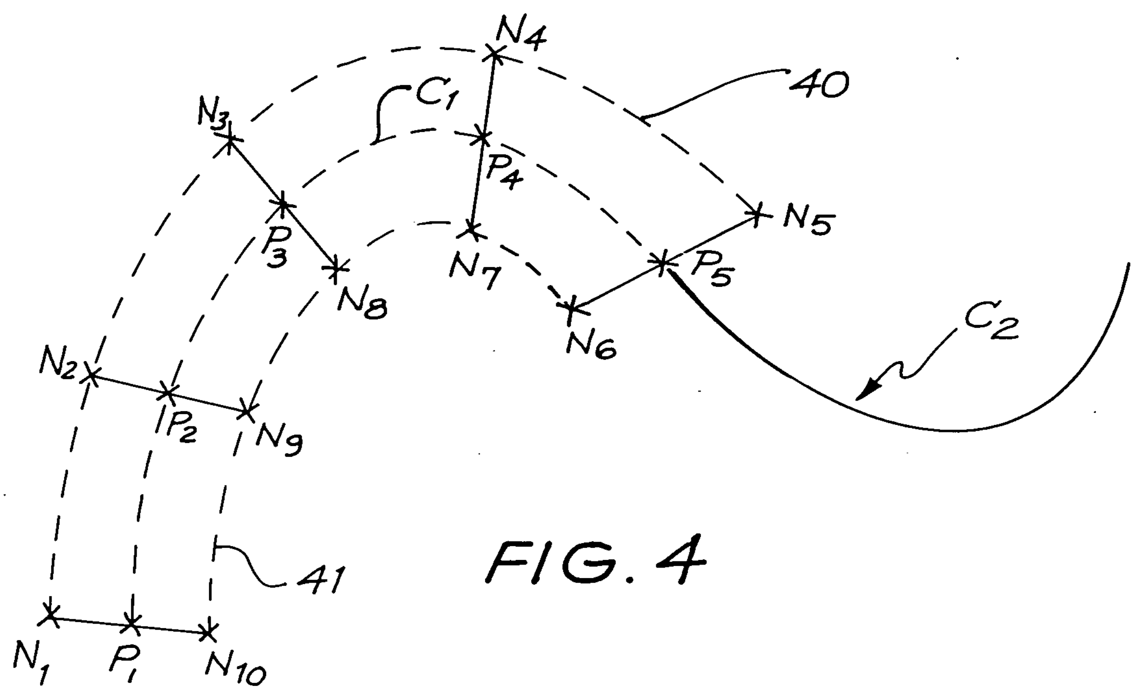
N4
40
C1
N3
P4
N5
P3
P5
N7
N8
N6
C2
N2
P2
N9
41
N1
P1
N10

FIG. 4

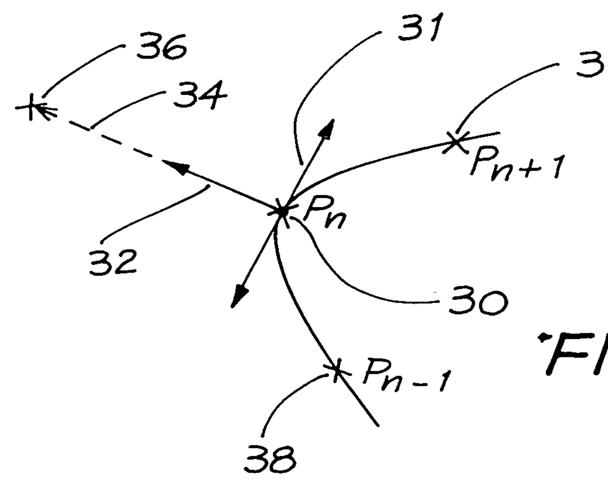
31
36
39
34
Pn+1
Pn
32
30
Pn-1
38

FIG. 3

Appendix A – Related Australian Provisional Patent Applications

The present provisional is one of a series of interrelated Australian Provisional Patent Applications filed concurrently by the present Applicant and which together relate to a new image processing system which presents a large number of significant advances in a number of technological fields. These fields include, but are not limited to those set out in the following table:

- Camera technologies
- Display technologies
- Image processing
- Ink Jet printing technology
- Semiconductor fabrication technology
- Micro Electro Mechanical Systems (MEMS)
- VLSI and ULSI fabrication including Thin Field Techology
- Magnetics
- Fluid dynamics
- Precision engineering
- Plastics molding
- Materials science
- Digital systems architecture
- Fluid Dynamics
- Precision Engineering
- Non-impact printing technologies
- Mechanical and stress analysis
- Ink Chemistry
- Electronics
- Electrostatics

Naturally with such a large number of significant advances, it is necessary to read this Application with its associated Australian Provisional Patent Applications to gain a thorough understanding of the operation of these technologies. The following tables set out a full list of the associated Australian Provisional Patent Applications filed concurrently herewith by the present applicant which should be referred to in obtaining a full understanding of the operation of the present invention:

Ink Jet Printing

A large number of new forms of ink jet printers have been developed to facilitate alternative ink jet technologies for the image processing system. Australian Provisional Patent Applications relating to these ink jets include:

Image Creation Method and Apparatus (IJ01)
Image Creation Method and Apparatus (IJ02)
Image Creation Method and Apparatus (IJ03)
Image Creation Method and Apparatus (IJ04)
Image Creation Method and Apparatus (IJ05)
Image Creation Method and Apparatus (IJ06)
Image Creation Method and Apparatus (IJ07)
Image Creation Method and Apparatus (IJ08)
Image Creation Method and Apparatus (IJ09)
Image Creation Method and Apparatus (IJ10)
Image Creation Method and Apparatus (IJ11)
Image Creation Method and Apparatus (IJ12)
Image Creation Method and Apparatus (IJ13)
Image Creation Method and Apparatus (IJ14)

Image Creation Method and Apparatus (IJ15)
Image Creation Method and Apparatus (IJ16)
Image Creation Method and Apparatus (IJ17)
Image Creation Method and Apparatus (IJ18)
Image Creation Method and Apparatus (IJ19)
Image Creation Method and Apparatus (IJ20)
Image Creation Method and Apparatus (IJ21)
Image Creation Method and Apparatus (IJ22)
Image Creation Method and Apparatus (IJ23)
Image Creation Method and Apparatus (IJ24)
Image Creation Method and Apparatus (IJ25)
Image Creation Method and Apparatus (IJ26)
Image Creation Method and Apparatus (IJ27)
Image Creation Method and Apparatus (IJ28)
Image Creation Method and Apparatus (IJ29)
Image Creation Method and Apparatus (IJ30)
Supply Method and Apparatus (F1)
Supply Method and Apparatus (F2)

Ink Jet Manufacturing

Significant developments have occurred in the field of ink jet print head construction. These advances are included in the following Australian Provisional Patent Applications.

A Method of Manufacture of an Image Creation Apparatus (IJM01)
A Method of Manufacture of an Image Creation Apparatus (IJM02)
A Method of Manufacture of an Image Creation Apparatus (IJM03)
A Method of Manufacture of an Image Creation Apparatus (IJM04)
A Method of Manufacture of an Image Creation Apparatus (IJM05)
A Method of Manufacture of an Image Creation Apparatus (IJM06)
A Method of Manufacture of an Image Creation Apparatus (IJM07)
A Method of Manufacture of an Image Creation Apparatus (IJM08)
A Method of Manufacture of an Image Creation Apparatus (IJM09)
A Method of Manufacture of an Image Creation Apparatus (IJM10)
A Method of Manufacture of an Image Creation Apparatus (IJM11)
A Method of Manufacture of an Image Creation Apparatus (IJM12)
A Method of Manufacture of an Image Creation Apparatus (IJM13)
A Method of Manufacture of an Image Creation Apparatus (IJM14)
A Method of Manufacture of an Image Creation Apparatus (IJM15)
A Method of Manufacture of an Image Creation Apparatus (IJM16)
A Method of Manufacture of an Image Creation Apparatus (IJM17)
A Method of Manufacture of an Image Creation Apparatus (IJM18)
A Method of Manufacture of an Image Creation Apparatus (IJM19)
A Method of Manufacture of an Image Creation Apparatus (IJM20)
A Method of Manufacture of an Image Creation Apparatus (IJM21)
A Method of Manufacture of an Image Creation Apparatus (IJM22)
A Method of Manufacture of an Image Creation Apparatus (IJM23)
A Method of Manufacture of an Image Creation Apparatus (IJM24)
A Method of Manufacture of an Image Creation Apparatus (IJM25)
A Method of Manufacture of an Image Creation Apparatus (IJM26)
A Method of Manufacture of an Image Creation Apparatus (IJM27)
A Method of Manufacture of an Image Creation Apparatus (IJM28)
A Method of Manufacture of an Image Creation Apparatus (IJM29)
A Method of Manufacture of an Image Creation Apparatus (IJM30)

MEMS Technology

The following application relate to Micro Electro-Mechanical Systems technologies:

A device (MEMS01)
A device (MEMS02)
A device (MEMS03)
A device (MEMS04)
A device (MEMS05)
A device (MEMS06)
A device (MEMS07)
A device (MEMS08)
A device (MEMS09)
A device (MEMS10)

Artcam Technologies

The following Australian Provisional Patent Applications relate to the a new field of image processing technology known as Artcam.

Image Processing Method and Apparatus (ART01)
Image Processing Method and Apparatus (ART02)
Image Processing Method and Apparatus (ART03)
Image Processing Method and Apparatus (ART05)
Image Processing Method and Apparatus (ART06)
Media Device (ART07)
Image Processing Method and Apparatus (ART08)
Image Processing Method and Apparatus (ART09)
Image Processing Method and Apparatus (ART10)
Image Processing Method and Apparatus (ART11)
Image Processing Method and Apparatus (ART12)
Media Device (ART13)
Image Processing Method and Apparatus (ART12)
Media Device (ART15)
Media Device (ART16)
Media Device (ART17)
Media Device (ART18)
Data Processing Method and Apparatus (ART19)
Data Processing Method and Apparatus (ART20)
Media Processing Method and Apparatus (ART21)
Image Processing Method and Apparatus (ART22)
Image Processing Method and Apparatus (ART23)
Image Processing Method and Apparatus (ART24)
Image Processing Method and Apparatus (ART25)
Image Processing Method and Apparatus (ART26)
Image Processing Method and Apparatus (ART27)
Data Processing Method and Apparatus (ART29)
Data Processing Method and Apparatus (ART32)
Image Processing Method and Apparatus (ART33)
Sensor Creation Method and Apparatus (ART36)
Data Processing Method and Apparatus (ART37)
Data Processing Method and Apparatus (ART38)
Data Processing Method and Apparatus (ART39)
Data Processing Method and Apparatus (ART40)
Data Processing Method and Apparatus (ART43)
Data Processing Method and Apparatus (ART44)
Data Processing Method and Apparatus (ART45)
Data Processing Method and Apparatus (ART46)

Data Processing Method and Apparatus (ART50)
Data Processing Method and Apparatus (ART51)
Data Processing Method and Apparatus (ART52)
Image Processing Method and Apparatus (ART53)
Image Processing Method and Apparatus (ART54)
Image Processing Method and Apparatus (ART56)